Exhibit K-2

	the next available segment. At that time, LFS sorts the blocks by logical block number, assigns them disk addresses, and updates the meta-data to reflect their addresses." Seltzer, Sec. 6.1.1, p. 71. Once some buffers holding dirty pages (changed data) are flushed to disk blocks, they, together with unchanged blocks extant on disk, constitute a second set of blocks. This second set of blocks, in combination with un-flushed dirty buffers, represent the up-to-date file system in a consistent state.
18. An article of manufacture as in claim 17, wherein the instructions further cause the processor to move atomically said file system on said storage system from said first consistent state to said second consistent state.	Seltzer teaches that the LFS file system atomically advances when the *ifile* inode is committed to disk. *See e.g.* Seltzer Sec. 6.1.1, pp. 71-72, Sec. 6.3.3, p. 88. A consistent checkpoint is created when the *ifile* inode is flushed to disk. *Id.*
19. An article of manufacture as in claim 18, wherein the instructions cause the processor to move atomically said file system on said storage system to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Seltzer teaches that the new *ifile* inode is flushed to disk after the dirty buffers are flushed. *See* Seltzer, Sec. 6.1.1, pp. 71-72, Sec. 6.3.3, p. 88.
20. An article of manufacture as in claim 19, wherein updating said on-disk root inode comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	Seltzer teaches that the superblock is replicated to allow recovery from crashes that corrupt the primary copy of the superblock. Seltzer, Sec. 6.1.1, p. 71. In view of this teaching, it would have been obvious for one of ordinary skill in the art to replicate the root inode to recover from crashes that corrupt the primary copy of the root inode.
21. An article of manufacture as in claim 17, wherein the instructions further cause the processor to create a snapshot of said file system by copying only said on-disk root inode.	Seltzer teaches that the LFS file system atomically advances when the *ifile* inode is committed to disk. *See e.g.* Seltzer Sec. 6.1.1, pp. 71-72, Sec. 6.3.3, p. 88. A consistent checkpoint is created when the *ifile* inode is flushed to disk. *Id.*
22. An article of manufacture as in claim 21, wherein the instructions cause the	Seltzer teaches that the LFS file system atomically advances when the *ifile* inode is committed to disk.

processor to create said snapshot so that said snapshot and said file system share said first set of blocks on said storage system when said snapshot is created.	*See e.g.* Seltzer Sec. 6.1.1, pp. 71-72, Sec. 6.3.3, p. 88. A consistent checkpoint is created when the *ifile* inode is flushed to disk. *Id.* The checkpoint inherently shares unmodified blocks with the active file system.

Fourth Basis of Invalidity

The reference applicable to the fourth basis of invalidity is:

1. Schilling, *Design and implementation of a fast file system for Unix with special consideration of technical parameters of optical storage media and multimedia applications*, Thesis submitted to Technical University of Berlin on 5/23/1991, translated from German. ("Schilling"). All pages cited are to the English translation.

The pertinence and manner of applying Schilling to claims 1-24 for which re-examination is requested is as follows:

Claims of '211 Patent	Schilling
1. A method of maintaining a file system stored in a memory and on a storage system that includes one or more hard disks, said method comprising steps of:	Schilling teaches a WoFS file system using a cache memory and one or more WORM optical disks. Schilling further teaches that the WoFS file system can be used with magnetic hard disks. Schilling, Sec. 1.2, p. 5.
maintaining an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system; and	Schilling teaches that the WoFS file system is based on the well-known Unix file structure (as implemented in industry-standard BSD file system), which includes a superblock (root inode), inodes with levels of indirection, and data blocks. Schilling, Sec. 1.2.2, p. 6. WoFS modifies this structure to account for the write once – read many (WORM) nature of the optical media. *See e.g.* Schilling, Sec. 12.3-1.2.4. The modified system retains superblocks, inodes (referred to as *generation* nodes or *gnodes*) and data blocks. *Id.* These structures are present on the WORM disks. *See* Schilling Sec. 1.2.5. WoFS uses copy-on-write techniques to achieve file system consistency, writing new versions of files to a previously unused area of the medium. *See e.g.* Schilling, Sec. 1.2.5.1, p. 12
maintaining an incore root inode in said memory, said incore root inode pointing	Schilling teaches a cache memory where an in-core copy of the on-disk file system structure is

directly and indirectly to buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	maintained. *See* Schilling, Sec. 1.4.2. Changes to the file system occur in the cached file system structure, which thereby diverges from the on-disk structure. *See* Schilling, Sec. 1.2.7. It is inherent that changed data is stored in buffers in the cache before being written to disk. Schilling teaches that gnode updates are periodically forced to disk. Schilling, Sec. 1.2.7.4, p. 20. It is inherent that updated gnodes are buffered in the cache before being written to disk. Schilling further teaches that the superblock is written to disk after data blocks and intermediate gnodes. Schilling, Sec. 1.2.7.2, p. 19. It is inherent that the updated superblock, pointing to buffered and on-disk gnodes and data blocks, is buffered in the cache before being forced to disk.
2. A method as in claim 1, wherein said file system on said storage system always moves atomically from said first consistent state to said second consistent state.	Schilling's WoFS system advances atomically when the superblock update is forced to disk. *See* Schilling, Sec. 1.2.7.2.
3. A method as in claim 2, wherein said file system on said storage system moves atomically to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Schilling's WoFS system advances atomically when the superblock update is forced to disk. *See* Schilling, Sec. 1.2.7.2. Schilling further teaches that the superblock is written to disk after data blocks and intermediate gnodes. Schilling, Sec. 1.2.7.2, p. 19.
4. A method as in claim 3, wherein updating said on-disk root inode further comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	Replicating the metadata, such as a root inode, is a technique that is well known in the art. Schilling teaches that the superblock is replicated to allow recovery from crashes that corrupt the primary copy of the superblock. Schilling, Sec. 1.2.7.1, p. 19.
5. A method as in claim 1, further comprising the step of creating a snapshot of said file system by copying only said on-disk root inode.	Schilling teaches that old versions of gnodes and associated data (equivalent to snapshots) are retained because WORM disks cannot be erased. Schilling, Sec. 1.2.5.1, p. 12. Schilling further teaches accessing old versions through a directory. *Id.* It is inherent that the directory would be a part of the root node or superblock

	and be copied when the root is copied.
6. A method as in claim 5, wherein when said snapshot is created, said snapshot and said file system share said first set of blocks on said storage system.	Due to the copy-on-write characteristic of the WoFS file system, older and newer versions of data would share unchanged data blocks.
7. A method as in claim 1, further comprising the step of creating plural snapshots of said file system by copying only said on-disk root inode at different times.	Schilling teaches that old versions of gnodes and associated data (equivalent to snapshots) are retained because WORM disks cannot be erased. Schilling, Sec. 1.2.5.1, p. 12. .Schilling further teaches accessing old versions through a directory. *Id.* It is inherent that the directory would be a part of the root node or superblock and be copied when the root is copied.
8. A method as in claim 7, wherein when each one of said plural snapshots is created, each one of said snapshots and said file system share said first set of blocks on said storage system.	Due to the copy-on-write characteristic of the WoFS file system, older and newer versions of data would share unchanged data blocks.
9. A device comprising: a processor; a memory; and	The file system taught in Schilling is inherently executed on a processor with a memory.
a storage system including one or more hard disks;	Schilling teaches a WoFS file system using a cache memory and one or more WORM optical disks. Schilling further teaches that the WoFS file system can be used with magnetic hard disks. Schilling, Sec. 1.2, p. 5
wherein said memory and said storage system store a file system; and	The file system in Schilling is stored in memory and on a storage system. *Id.*
wherein said memory also stores information including instructions executable by said processor to maintain said file system, the instructions including steps of (a) maintaining an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system, and (b) maintaining an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said	Schilling teaches that the WoFS file system is based on the well-known Unix file structure (as implemented in industry-standard BSD file system), which includes a superblock (root inode), inodes with levels of indirection, and data blocks. Schilling, Sec. 1.2.2, p. 6. WoFS modifies this structure to account for the write once – read many (WORM) nature of the optical media. *See e.g.* Schilling, Sec. 12.3-1.2.4. The modified system retains superblocks, inodes (referred to as *generation* nodes or *gnodes*) and data blocks. *Id.* These structures are present on the WORM disks. *See* Schilling Sec. 1.2.5. WoFS uses copy-on-write techniques to achieve file system

second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	consistency, writing new versions of files to a previously unused area of the medium. *See e.g.* Schilling, Sec. 1.2.5.1, p. 12 Schilling further teaches a cache memory where an in-core copy of the on-disk file system structure is maintained. *See* Schilling, Sec. 1.4.2. Changes to the file system occur in the cached file system structure, which thereby diverges from the on-disk structure. *See* Schilling, Sec. 1.2.7. It is inherent that changed data is stored in buffers in the cache before being written to disk. Schilling teaches that gnode updates are periodically forced to disk. Schilling, Sec. 1.2.7.4, p. 20. It is inherent that updated gnodes are buffered in the cache before being written to disk. Schilling further teaches that the superblock is written to disk after data blocks and intermediate gnodes. Schilling, Sec. 1.2.7.2, p. 19. It is inherent that the updated superblock, pointing to buffered and on-disk gnodes and data blocks, is buffered in the cache before being forced to disk.
10. A device as in claim 9, wherein said file system on said storage system always moves atomically from said first consistent state to said second consistent state.	Schilling's WoFS system advances atomically when the superblock update is forced to disk. *See* Schilling, Sec. 1.2.7.2.
11. A device as in claim 10, wherein said file system on said storage system moves atomically to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Schilling's WoFS system advances atomically when the superblock update is forced to disk. *See* Schilling, Sec. 1.2.7.2. Schilling further teaches that the superblock is written to disk after data blocks and intermediate gnodes. Schilling, Sec. 1.2.7.2, p. 19.
12. A device as in claim 11, wherein updating said on-disk root inode further comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	Replicating the metadata, such as a root inode, is a technique that is well known in the art. Schilling teaches that the superblock is replicated to allow recovery from crashes that corrupt the primary copy of the superblock. Schilling, Sec. 1.2.7.1, p. 19.

13. A device as in claim 9, wherein the instructions further comprise the step of creating a snapshot of said file system by copying only said on-disk root inode.	Schilling teaches that old versions of gnodes and associated data (equivalent to snapshots) are retained because WORM disks cannot be erased. Schilling, Sec. 1.2.5.1, p. 12. Schilling further teaches accessing old versions through a directory. *Id.* It is inherent that the directory would be a part of the root node or superblock and be copied when the root is copied.
14. A device as in claim 13, wherein when said snapshot is created, said snapshot and said file system share said first set of blocks on said storage system.	Due to the copy-on-write characteristic of the WoFS file system, older and newer versions of data would share unchanged data blocks.
15. A device as in claim 9, wherein the instructions further comprise the step of creating plural snapshots of said file system by copying only said on-disk root inode at different times.	Schilling teaches that old versions of gnodes and associated data (equivalent to snapshots) are retained because WORM disks cannot be erased. Schilling, Sec. 1.2.5.1, p. 12. Schilling further teaches accessing old versions through a directory. *Id.* It is inherent that the directory would be a part of the root node or superblock and be copied when the root is copied.
16. A device as in claim 15, wherein when each one of said plural snapshots is created, each one of said snapshots and said file system share said first set of blocks on said storage system.	Due to the copy-on-write characteristic of the WoFS file system, older and newer versions of data would share unchanged data blocks.
17. An article of manufacture comprising a machine-readable storage medium storing instructions executable by a processor coupled to a memory and to a storage system, said storage system comprising one or more hard disks, said memory and said storage system storing a file system, wherein the instructions, when executed by the processor, cause the processor to (a) maintain an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system, and (b) maintain an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system,	The file system of Schilling is inherently stored on machine readable medium in the form of instructions executable by a processor coupled to a memory and one or more hard disks. Schilling teaches that the WoFS file system is based on the well-known Unix file structure (as implemented in industry-standard BSD file system), which includes a superblock (root inode), inodes with levels of indirection, and data blocks. Schilling, Sec. 1.2.2, p. 6. WoFS modifies this structure to account for the write once – read many (WORM) nature of the optical media. *See e.g.* Schilling, Sec. 12.3-1.2.4. The modified system retains superblocks, inodes (referred to as *generation* nodes or *gnodes*) and data blocks. *Id.* These

said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	structures are present on the WORM disks. *See* Schilling Sec. 1.2.5. WoFS uses copy-on-write techniques to achieve file system consistency, writing new versions of files to a previously unused area of the medium. *See e.g.* Schilling, Sec. 1.2.5.1, p. 12 Schilling further teaches a cache memory where an in-core copy of the on-disk file system structure is maintained. *See* Schilling, Sec. 1.4.2. Changes to the file system occur in the cached file system structure, which thereby diverges from the on-disk structure. *See* Schilling, Sec. 1.2.7. It is inherent that changed data is stored in buffers in the cache before being written to disk. Schilling teaches that gnode updates are periodically forced to disk. Schilling, Sec. 1.2.7.4, p. 20. It is inherent that updated gnodes are buffered in the cache before being written to disk. Schilling further teaches that the superblock is written to disk after data blocks and intermediate gnodes. Schilling, Sec. 1.2.7.2, p. 19. It is inherent that the updated superblock, pointing to buffered and on-disk gnodes and data blocks, is buffered in the cache before being forced to disk.
18. An article of manufacture as in claim 17, wherein the instructions further cause the processor to move atomically said file system on said storage system from said first consistent state to said second consistent state.	Schilling's WoFS system advances atomically when the superblock update is forced to disk. *See* Schilling, Sec. 1.2.7.2.
19. An article of manufacture as in claim 18, wherein the instructions cause the processor to move atomically said file system on said storage system to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Schilling's WoFS system advances atomically when the superblock update is forced to disk. *See* Schilling, Sec. 1.2.7.2. Schilling further teaches that the superblock is written to disk after data blocks and intermediate gnodes. Schilling, Sec. 1.2.7.2, p. 19.
20. An article of manufacture as in claim 19, wherein updating said on-disk root inode comprises updating said on-disk root inode and	Replicating the metadata, such as a root inode, is a technique that is well known in the art. Schilling teaches that the superblock is

then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	replicated to allow recovery from crashes that corrupt the primary copy of the superblock. Schilling, Sec. 1.2.7.1, p. 19.
21. An article of manufacture as in claim 17, wherein the instructions further cause the processor to create a snapshot of said file system by copying only said on-disk root inode.	Schilling teaches that old versions of gnodes and associated data (equivalent to snapshots) are retained because WORM disks cannot be erased. Schilling, Sec. 1.2.5.1, p. 12. Schilling further teaches accessing old versions through a directory. *Id.* It is inherent that the directory would be a part of the root node or superblock and be copied when the root is copied.
22. An article of manufacture as in claim 21, wherein the instructions cause the processor to create said snapshot so that said snapshot and said file system share said first set of blocks on said storage system when said snapshot is created.	Due to the copy-on-write characteristic of the WoFS file system, older and newer versions of data would share unchanged data blocks.
23. An article of manufacture as in claim 17, wherein the instructions further cause the processor to create plural snapshots of said file system by copying only said on-disk root inode at different times.	Schilling teaches that old versions of gnodes and associated data (equivalent to snapshots) are retained because WORM disks cannot be erased. Schilling, Sec. 1.2.5.1, p. 12. Schilling further teaches accessing old versions through a directory. *Id.* It is inherent that the directory would be a part of the root node or superblock and be copied when the root is copied.
24. An article of manufacture as in claim 23, wherein the instructions, when executed by the processor, cause the processor to create each one of said plural snapshots so that each one of said snapshots and said file system share said first set of blocks on said storage system when each one of said plural snapshots is created.	Due to the copy-on-write characteristic of the WoFS file system, older and newer versions of data would share unchanged data blocks.

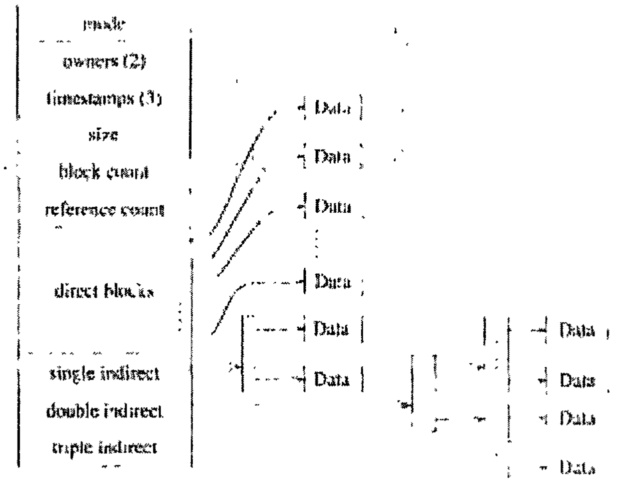
<u>Fifth Basis of Invalidity</u>

The references applicable to the fifth basis of invalidity are:

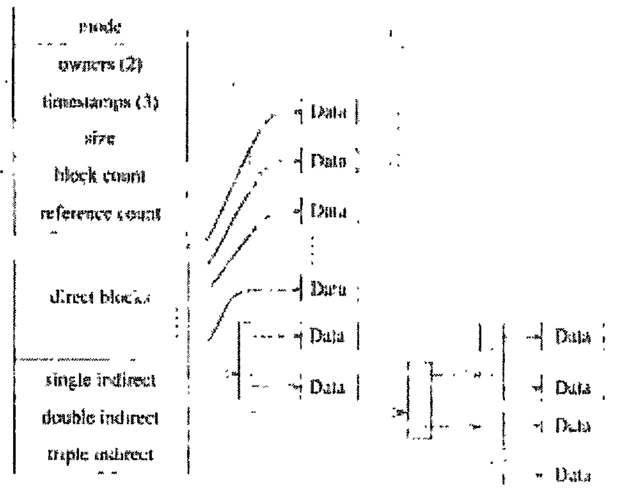
1. Leffler, McKusick, et. al., *4.3BSD Unix Operating System*, Addison-Wesley Publishing Co., 1990 ("Leffler").

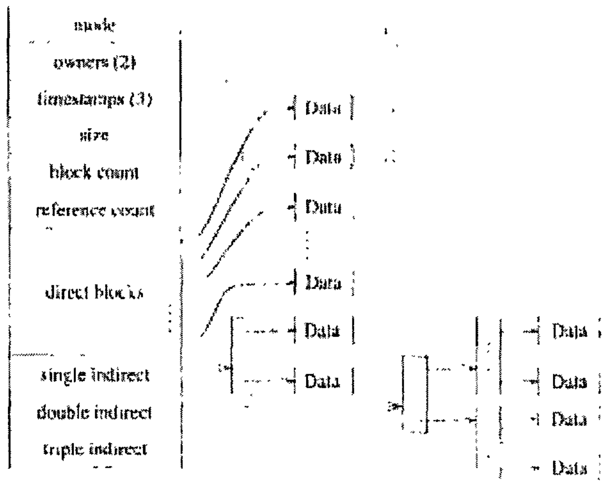
2. Bach, *The Design of the Unix Operating System*, Prentice Hall, 1990 ("Bach").

The pertinence and manner of applying Leffler and Bach to claims 1-4, 9-12, 17-20 for which re-examination is requested is as follows:

Claims of '211 Patent	Leffler and Bach
1. A method of maintaining a file system stored in a memory and on a storage system that includes one or more hard disks, said method comprising steps of:	Leffler teaches an implementation of the well-known Unix file system, storing information in memory and on hard disk. *See* Leffler, Ch. 7.
maintaining an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system; and	Leffler teaches maintaining an on-disk root inode pointing directly and indirectly to a set of blocks on disk in a consistent state: **Figure 7.6** The structure of an inode.  *See* Leffler Sec. 7.2
maintaining an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	Leffler teaches that a copy of the root inode (inode describing the root directory) and copies of lower level inodes and data blocks are maintained in memory (in-core). Leffler, Sec. 7.4, pp. 203-07 When the system writes new data, it allocates buffers in memory for the new data. *See e.g.* Leffler, Sec. 7.5, pp. 211-12. Periodically, the buffers are flushed to disk during a *sync* process. *See e.g.* Leffler, Sec. 7.4, p. 207. It is inherent that the incore root inode is updated to point to the new buffers and blocks. For instance, Bach (a conventional well known Unix textbook) teaches that in Unix, the in-core inode is updated when data blocks change. Bach, Sec. 4.1, pp. 62-63.

2. A method as in claim 1, wherein said file system on said storage system always moves atomically from said first consistent state to said second consistent state.	Leffler teaches that the file system advances during the periodic *sync* process when dirty buffers, including root inode, are forced to disk. *See* Leffler Sec. 7.4, pp.206-07; *see also e.g.* Bach, Sec. 4.1.3, p. 67.
3. A method as in claim 2, wherein said file system on said storage system moves atomically to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Leffler teaches that the file system advances during the periodic *sync* process when dirty buffers, including root inode, are forced to disk. *See* Leffler Sec. 7.4, pp.206-07; *see also e.g.* Bach, Sec. 4.1.3, p. 67.
4. A method as in claim 3, wherein updating said on-disk root inode further comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	Replicating the metadata, such as a root inode, is a technique that is well known in the art. Leffler teaches that in the Unix filesystem the superblock is replicated to allow recovery from crashes that corrupt the primary copy of the superblock. Leffler, Sec. 7.3, p. 196. In view of the knowledge in the art and Leffler's teaching, it would have been obvious for one of ordinary skill to make a copy of the root inode to recover from crashes that corrupt the primary copy of the root inode.
9. A device comprising: a processor; a memory; and	The file system taught in Leffler is inherently executed on a processor with a memory.
a storage system including one or more hard disks;	Leffler teaches a file system using a cache memory and one or more disks.
wherein said memory and said storage system store a file system; and	The file system in Leffler is stored in memory and on a storage system.
wherein said memory also stores information including instructions executable by said processor to maintain said file system, the instructions including steps of (a) maintaining an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system, and (b) maintaining an incore root inode in said memory, said incore root inode pointing directly and indirectly to	Leffler teaches maintaining an on-disk root inode pointing directly and indirectly to a set of blocks on disk in a consistent state:

buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	**Figure 7.6** The structure of an inode.  inode owners (2) timestamps (3) size block count reference count direct blocks single indirect double indirect triple indirect Data *See* Leffler Sec. 7.2 Leffler further teaches that a copy of the root inode (inode describing the root directory) and copies of lower level inodes and data blocks are maintained in memory (in-core). Leffler, Sec. 7.4, pp. 203-07 When the system writes new data, it allocates buffers in memory for the new data. *See e.g.* Leffler, Sec. 7.5, pp. 211-12. Periodically, the buffers are flushed to disk during a *sync* process. *See e.g.* Leffler, Sec. 7.4, p. 207. It is inherent that the incore root inode is updated to point to the new buffers and blocks. For instance, Bach (a conventional well known Unix textbook) teaches that in Unix, the in-core inode is updated when data blocks change. Bach, Sec. 4.1, pp. 62-63.
10. A device as in claim 9, wherein said file system on said storage system always moves atomically from said first consistent state to said second consistent state.	Leffler teaches that the file system advances during the periodic *sync* process when dirty buffers, including root inode, are forced to disk. *See* Leffler Sec. 7.4, pp.206-07; *see also e.g.* Bach, Sec. 4.1.3, p. 67.
11. A device as in claim 10, wherein said file system on said storage system moves atomically to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Leffler teaches that the file system advances during the periodic *sync* process when dirty buffers, including root inode, are forced to disk. *See* Leffler Sec. 7.4, pp.206-07; *see also e.g.* Bach, Sec. 4.1.3, p. 67.
12. A device as in claim 11, wherein	Replicating the metadata, such as a root inode, is a

updating said on-disk root inode further comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	technique that is well known in the art. Leffler teaches that in the Unix filesystem the superblock is replicated to allow recovery from crashes that corrupt the primary copy of the superblock. Leffler, Sec. 7.3, p. 196. In view of the knowledge in the art and Leffler's teaching, it would have been obvious for one of ordinary skill to make a copy of the root inode to recover from crashes that corrupt the primary copy of the root inode.
17. An article of manufacture comprising a machine-readable storage medium storing instructions executable by a processor coupled to a memory and to a storage system, said storage system comprising one or more hard disks, said memory and said storage system storing a file system, wherein the instructions, when executed by the processor, cause the processor to (a) maintain an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system, and (b) maintain an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	The file system in Leffler is inherently stored on machine readable medium in the form of instructions executable by a processor coupled to a memory and one or more hard disks. Leffler teaches maintaining an on-disk root inode pointing directly and indirectly to a set of blocks on disk in a consistent state: **Figure 7.8** The structure of an inode.  inode owners (2) timestamps (3) size block count reference count direct blocks single indirect double indirect triple indirect Data Data Data Data Data Data Data Data Data Data *See* Leffler Sec. 7.2 Leffler further teaches that a copy of the root inode (inode describing the root directory) and copies of lower level inodes and data blocks are maintained in memory (in-core). Leffler, Sec. 7.4, pp. 203-07 When the system writes new data, it allocates buffers in memory for the new data. *See e.g.* Leffler, Sec. 7.5, pp. 211-12. Periodically, the buffers are flushed to disk during a *sync* process. *See e.g.* Leffler, Sec. 7.4, p. 207. It is inherent that the incore root inode is updated to point to the new buffers and blocks. For instance, Bach (a conventional well known Unix textbook) teaches that in Unix, the in-core inode is updated when data blocks change. Bach, Sec. 4.1, pp. 62-63.

18. An article of manufacture as in claim 17, wherein the instructions further cause the processor to move atomically said file system on said storage system from said first consistent state to said second consistent state.	Leffler teaches that the file system advances during the periodic *sync* process when dirty buffers, including root inode, are forced to disk. *See* Leffler Sec. 7.4, pp.206-07; *see also e.g.* Bach, Sec. 4.1.3, p. 67.
19. An article of manufacture as in claim 18, wherein the instructions cause the processor to move atomically said file system on said storage system to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Leffler teaches that the file system advances during the periodic *sync* process when dirty buffers, including root inode, are forced to disk. *See* Leffler Sec. 7.4, pp.206-07; *see also e.g.* Bach, Sec. 4.1.3, p. 67.
20. An article of manufacture as in claim 19, wherein updating said on-disk root inode comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	Replicating the metadata, such as a root inode, is a technique that is well known in the art. Leffler teaches that in the Unix filesystem the superblock is replicated to allow recovery from crashes that corrupt the primary copy of the superblock. Leffler, Sec. 7.3, p. 196. In view of the knowledge in the art and Leffler's teaching, it would have been obvious for one of ordinary skill to make a copy of the root inode to recover from crashes that corrupt the primary copy of the root inode.

Sixth Basis of Invalidity

The references applicable to the sixth basis of invalidity are:

1. Rosenblum, Ousterhoot, *The LFS Storage Manager*, presented at USENIX Tech. Conf., Anaheim, CA, 1990 ("Rosenblum").

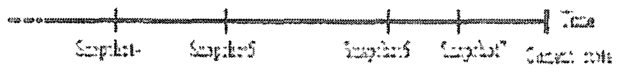
2. Ylonen, as cited hereinabove.

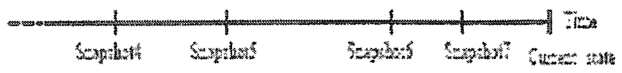
3. Leffler, as cited hereinabove.

The pertinence and manner of applying Rosenblum, Ylonen and Leffler claims 1-24 for which re-examination is requested is as follows:

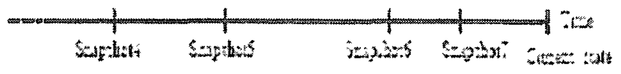
Claims of '211 Patent	Rosenblum, Ylonen and Leffler

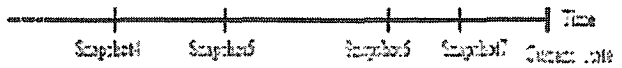
1. A method of maintaining a file system stored in a memory and on a storage system that includes one or more hard disks, said method comprising steps of:	Rosenblum teaches a file system stored in memory and on hard disks.
maintaining an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system; and	Rosenblum maintains an on disk root inode. The format of the inodes and indirect blocks is the same as in the standard Unix file system, which is known to one of ordinary skill to provide for root inodes that directly and indirectly point to data blocks. *See* Rosenblum, Sec. 4.2.1, p.6.
maintaining an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	Rosenblum teaches that file system metadata, including inodes, and data blocks are cached in memory, where changes are accumulated before being forced to disk. Rosenblum, Sec. 4.1, p. 5. Before being forced to disk, the modified in-core root inode inherently points to buffers in memory containing modified data, new blocks already forced to disk, and old blocks with unmodified data.
2. A method as in claim 1, wherein said file system on said storage system always moves atomically from said first consistent state to said second consistent state.	Rosenblum teaches a checkpoint, which marks a consistent state of the file system. Rosenblum, Sec. 4.4.1, p. 9. Once all modifications are written to disk, a checkpoint region is written, atomically moving the file system to a new consistent state. *Id.* The checkpoint region points to the last segment written and the location of the inode map. *Id.*
3. A method as in claim 2, wherein said file system on said storage system moves atomically to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Rosenblum teaches a checkpoint, which marks a consistent state of the file system. Rosenblum, Sec. 4.4.1, p. 9. Once all modifications are written to disk, a checkpoint region is written, atomically moving the file system to a new consistent state. *Id.* The checkpoint region points to the last segment written and the location of the inode map. *Id.*
4. A method as in claim 3, wherein	Replicating the metadata, such as a root inode, is a

updating said on-disk root inode further comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	technique that is well known in the art. For instance, Leffler teaches that in the Unix filesystem the superblock is replicated to allow recovery from crashes that corrupt the primary copy of the superblock. Leffler, Sec. 7.3, p. 196. In view of the knowledge in the art and, it would have been obvious for one of ordinary skill to make a copy of the root inode to recover from crashes that corrupt the primary copy of the root inode.
5. A method as in claim 1, further comprising the step of creating a snapshot of said file system by copying only said on-disk root inode.	Rosenblum teaches that the group commit concept at the basis of atomic updating taught in Rosenblum is found database literature. Rosenblum, Sec. 4.5, p. 10. Therefore, one of ordinary skill in the art would find motivation to look to the teachings of database art, such as Ylonen. Ylonen teaches an atomic commit method for databases, wherein a page table pointer (equivalent to a root inode) is updated to atomically advance the database to the next consistent state. Ylonen, Sec. 2, pp. 1-2, Fig. 1. Ylonen further teaches taking one or more snapshots by saving the address of the page table and preventing freeing of data pages. Ylonen, Sec. 8, p. 4, Fig. 2. The page table address is in the page table pointer. Thus, every time the page table pointer is copied (an operation that occurs at every commit cycle), "[w]e can have an arbitrary number of snapshots at different times, and they will all be consistent as long as their pages are not freed." *Id.*
6. A method as in claim 5, wherein when said snapshot is created, said snapshot and said file system share said first set of blocks on said storage system.	Ylonen teaches a shadow paging (copy-on-write) technique that "never modifies accessible pages," therefore a snapshot and the active file system share unmodified blocks. Ylonen, Sec. 8, p. 4, Figs. 1 and 2.  Figure 2. Snapshots represent consistent database states as of some time in the past.
7. A method as in claim 1, further comprising the step of creating plural snapshots of said file system by copying only said on-disk root inode at different times.	Ylonen teaches an atomic commit method for databases, wherein a page table pointer (equivalent to a root inode) is updated to atomically advance the database to the next consistent state. Ylonen, Sec. 2, pp. 1-2, Fig. 1. Ylonen further teaches taking one or

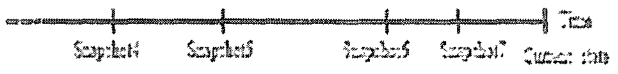
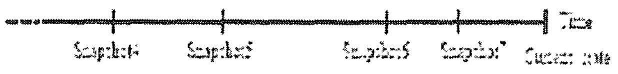
	more snapshots by saving the address of the page table and preventing freeing of data pages. Ylonen, Sec. 8, p. 4, Fig. 2. The page table address is in the page table pointer. Thus, every time the page table pointer is copied (an operation that occurs at every commit cycle), "[w]e can have an arbitrary number of snapshots at different times, and they will all be consistent as long as their pages are not freed." *Id.*
8. A method as in claim 7, wherein when each one of said plural snapshots is created, each one of said snapshots and said file system share said first set of blocks on said storage system.	Ylonen teaches a shadow paging (copy-on-write) technique that "never modifies accessible pages," therefore a snapshot and the active file system share unmodified blocks. Ylonen, Sec. 8, p. 4, Figs. 1 and 2.  Time Snapshot4 Snapshot5 Snapshot6 Snapshot7 Current state Figure 2: Snapshots represent consistent database states as of some time in the past.
9. A device comprising: a processor; a memory; and	The file system taught in Rosenblum is inherently executed on a processor with a memory.
a storage system including one or more hard disks;	Rosenblum teaches a file system using a cache memory and one or more disks.
wherein said memory and said storage system store a file system; and	The file system in Rosenblum is stored in memory and on a storage system.
wherein said memory also stores information including instructions executable by said processor to maintain said file system, the instructions including steps of (a) maintaining an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system, and (b) maintaining an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file	Rosenblum maintains an on disk root inode. The format of the inodes and indirect blocks is the same as in the standard Unix file system, which is known to one of ordinary skill to provide for root inodes that directly and indirectly point to data blocks. *See* Rosenblum, Sec. 4.2.1, p.6. Rosenblum further teaches that file system metadata, including inodes, and data blocks are cached in memory, where changes are accumulated before being forced to disk. Rosenblum, Sec. 4.1, p. 5. Before being forced to disk, the modified in-core root inode inherently points to buffers in memory containing modified data, new blocks already forced to disk, and old blocks with unmodified data.

system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	
10. A device as in claim 9, wherein said file system on said storage system always moves atomically from said first consistent state to said second consistent state.	Rosenblum teaches a checkpoint, which marks a consistent state of the file system. Rosenblum, Sec. 4.4.1, p. 9. Once all modifications are written to disk, a checkpoint region is written, atomically moving the file system to a new consistent state. *Id.* The checkpoint region points to the last segment written and the location of the inode map. *Id.*
11. A device as in claim 10, wherein said file system on said storage system moves atomically to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Rosenblum teaches a checkpoint, which marks a consistent state of the file system. Rosenblum, Sec. 4.4.1, p. 9. Once all modifications are written to disk, a checkpoint region is written, atomically moving the file system to a new consistent state. *Id.* The checkpoint region points to the last segment written and the location of the inode map. *Id.*
12. A device as in claim 11, wherein updating said on-disk root inode further comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	Replicating the metadata, such as a root inode, is a technique that is well known in the art. For instance, Leffler teaches that in the Unix filesystem the superblock is replicated to allow recovery from crashes that corrupt the primary copy of the superblock. Leffler, Sec. 7.3, p. 196. In view of the knowledge in the art and, it would have been obvious for one of ordinary skill to make a copy of the root inode to recover from crashes that corrupt the primary copy of the root inode.
13. A device as in claim 9, wherein the instructions further comprise the step of creating a snapshot of said file system by copying only said on-disk root inode.	Rosenblum teaches that the group commit concept at the basis of atomic updating taught in Rosenblum is found database literature. Rosenblum, Sec. 4.5, p. 10. Therefore, one of ordinary skill in the art would find motivation to look to the teachings of database art, such as Ylonen. Ylonen teaches an atomic commit method for databases, wherein a page table pointer (equivalent to a root inode) is updated to atomically advance the database to the next consistent state. Ylonen, Sec. 2,

	pp. 1-2, Fig. 1. Ylonen further teaches taking one or more snapshots by saving the address of the page table and preventing freeing of data pages. Ylonen, Sec. 8, p. 4, Fig. 2. The page table address is in the page table pointer. Thus, every time the page table pointer is copied (an operation that occurs at every commit cycle), "[w]e can have an arbitrary number of snapshots at different times, and they will all be consistent as long as their pages are not freed." *Id.*
14. A device as in claim 13, wherein when said snapshot is created, said snapshot and said file system share said first set of blocks on said storage system.	Ylonen teaches a shadow paging (copy-on-write) technique that "never modifies accessible pages," therefore a snapshot and the active file system share unmodified blocks. Ylonen, Sec. 8, p. 4, Figs. 1 and 2.  Figure 2: Snapshots represent consistent database states as of some time in the past.
15. A device as in claim 9, wherein the instructions further comprise the step of creating plural snapshots of said file system by copying only said on-disk root inode at different times.	Ylonen teaches an atomic commit method for databases, wherein a page table pointer (equivalent to a root inode) is updated to atomically advance the database to the next consistent state. Ylonen, Sec. 2, pp. 1-2, Fig. 1. Ylonen further teaches taking one or more snapshots by saving the address of the page table and preventing freeing of data pages. Ylonen, Sec. 8, p. 4, Fig. 2. The page table address is in the page table pointer. Thus, every time the page table pointer is copied (an operation that occurs at every commit cycle), "[w]e can have an arbitrary number of snapshots at different times, and they will all be consistent as long as their pages are not freed." *Id.*
16. A device as in claim 15, wherein when each one of said plural snapshots is created, each one of said snapshots and said file system share said first set of blocks on said storage system.	Ylonen teaches a shadow paging (copy-on-write) technique that "never modifies accessible pages," therefore a snapshot and the active file system share unmodified blocks. Ylonen, Sec. 8, p. 4, Figs. 1 and 2.

	 Time Snapshot4 Snapshot5 Snapshot6 Snapshot7 Current state Figure 2: Snapshots represent consistent database states as of some time in the past.
17. An article of manufacture comprising a machine-readable storage medium storing instructions executable by a processor coupled to a memory and to a storage system, said storage system comprising one or more hard disks, said memory and said storage system storing a file system, wherein the instructions, when executed by the processor, cause the processor to (a) maintain an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system, and (b) maintain an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	The file system in Rosenblum is inherently stored on machine readable medium in the form of instructions executable by a processor coupled to a memory and one or more hard disks. Rosenblum maintains an on disk root inode. The format of the inodes and indirect blocks is the same as in the standard Unix file system, which is known to one of ordinary skill to provide for root inodes that directly and indirectly point to data blocks. *See* Rosenblum, Sec. 4.2.1, p.6. Rosenblum further teaches that file system metadata, including inodes, and data blocks are cached in memory, where changes are accumulated before being forced to disk. Rosenblum, Sec. 4.1, p. 5. Before being forced to disk, the modified in-core root inode inherently points to buffers in memory containing modified data, new blocks already forced to disk, and old blocks with unmodified data.
18. An article of manufacture as in claim 17, wherein the instructions further cause the processor to move atomically said file system on said storage system from said first consistent state to said second	Rosenblum teaches a checkpoint, which marks a consistent state of the file system. Rosenblum, Sec. 4.4.1, p. 9. Once all modifications are written to disk, a checkpoint region is written, atomically moving the file system to a new consistent state. *Id.* The checkpoint region points to the last segment written and the

consistent state.	location of the inode map. *Id.*
19. An article of manufacture as in claim 18, wherein the instructions cause the processor to move atomically said file system on said storage system to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Rosenblum teaches a checkpoint, which marks a consistent state of the file system. Rosenblum, Sec. 4.4.1, p. 9. Once all modifications are written to disk, a checkpoint region is written, atomically moving the file system to a new consistent state. *Id.* The checkpoint region points to the last segment written and the location of the inode map. *Id.*
20. An article of manufacture as in claim 19, wherein updating said on-disk root inode comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	Replicating the metadata, such as a root inode, is a technique that is well known in the art. For instance, Leffler teaches that in the Unix filesystem the superblock is replicated to allow recovery from crashes that corrupt the primary copy of the superblock. Leffler, Sec. 7.3, p. 196. In view of the knowledge in the art and, it would have been obvious for one of ordinary skill to make a copy of the root inode to recover from crashes that corrupt the primary copy of the root inode.
21. An article of manufacture as in claim 17, wherein the instructions further cause the processor to create a snapshot of said file system by copying only said on-disk root inode.	Rosenblum teaches that the group commit concept at the basis of atomic updating taught in Rosenblum is found database literature. Rosenblum, Sec. 4.5, p. 10. Therefore, one of ordinary skill in the art would find motivation to look to the teachings of database art, such as Ylonen. Ylonen teaches an atomic commit method for databases, wherein a page table pointer (equivalent to a root inode) is updated to atomically advance the database to the next consistent state. Ylonen, Sec. 2, pp. 1-2, Fig. 1. Ylonen further teaches taking one or more snapshots by saving the address of the page table and preventing freeing of data pages. Ylonen, Sec. 8, p. 4, Fig. 2. The page table address is in the page table pointer. Thus, every time the page table pointer is copied (an operation that occurs at every commit cycle), "[w]e can have an arbitrary number of snapshots at different times, and they will all be consistent as long as their pages are not freed." *Id.*
22. An article of manufacture as in claim 21, wherein the instructions cause the processor to create said	Ylonen teaches a shadow paging (copy-on-write) technique that "never modifies accessible pages," therefore a snapshot and the active file system share

snapshot so that said snapshot and said file system share said first set of blocks on said storage system when said snapshot is created.	unmodified blocks. Ylonen, Sec. 8, p. 4, Figs. 1 and 2.  Figure 2: Snapshots represent consistent database states as of some time in the past.
23. An article of manufacture as in claim 17, wherein the instructions further cause the processor to create plural snapshots of said file system by copying only said on-disk root inode at different times.	Ylonen teaches an atomic commit method for databases, wherein a page table pointer (equivalent to a root inode) is updated to atomically advance the database to the next consistent state. Ylonen, Sec. 2, pp. 1-2, Fig. 1. Ylonen further teaches taking one or more snapshots by saving the address of the page table and preventing freeing of data pages. Ylonen, Sec. 8, p. 4, Fig. 2. The page table address is in the page table pointer. Thus, every time the page table pointer is copied (an operation that occurs at every commit cycle), "[w]e can have an arbitrary number of snapshots at different times, and they will all be consistent as long as their pages are not freed." *Id.*
24. An article of manufacture as in claim 23, wherein the instructions, when executed by the processor, cause the processor to create each one of said plural snapshots so that each one of said snapshots and said file system share said first set of blocks on said storage system when each one of said plural snapshots is created.	Ylonen teaches a shadow paging (copy-on-write) technique that "never modifies accessible pages," therefore a snapshot and the active file system share unmodified blocks. Ylonen, Sec. 8, p. 4, Figs. 1 and 2.  Figure 2: Snapshots represent consistent database states as of some time in the past.

<u>Other References</u>

1. Rosenblum, Ousterhoot, *The Design and Implementation of a Log-Structured File System*, Proceedings of the 13th ACM Symposium on OS Principles, 1991 ("Rosenblum and Ousterhoot").

Rosenblum and Ousterhoot disclosure is substantially similar to the Rosenblum reference cited hereinabove. Therefore, it anticipates or renders obvious the claims of the '211 patent for substantially the same reasons as the Rosenblum reference.

2. Kent, *Performance and Implementation Issues in Database Crash Recovery*, Ph.D. Dissertation, Princeton University, 1985 ("Kent").

Kent discloses a database using page shadowing (copy-on-write) and atomic commit to advance the database from one consistent state to another. Kent, Sec. 3.5, pp. 28-31, 40-43. Database structures include the root, which points to the global version of each mapping page. Kent, p. 41. "That is, the root defines the most recent, consistent state of the page table (Just as the page table defined the most recent, consistent state of the logical database.)" Kent, p. 41, Fig. 3-4. The structures shown in Fig. 3-4 would be understood by one of ordinary skill in the art to be equivalent to root inode, intermediate inodes, and data blocks. These structures reside on disk and in-core in cache memory. *See e.g.* Kent, p. 89-90. The cache contains buffers that hold modified data blocks. Kent, p. 90. Kent teaches an atomic commit operation by flushing the root from cache memory to disk. Kent, p. 42. "To commit, a transaction first flushes the subtree that corresponds to its modifications, and then the root." For at least these reasons, Kent anticipates or renders obvious at least claims 1-3, 9-11, and 17-19. Furthermore, as Fig. 3-4 shows, the root points to previous consistent states of the database, i.e. snapshots. So long as the older versions of data and metadata pages are not released, snapshots are accessible through the root and are created by copying and updating the root. Therefore, Kent anticipates or renders obvious claims 5-8, 13-16, and 21-24. Snapshots and multiple snapshots are further discussed in the Ilonen reference, cited hereinabove. Ilonen cites to Kent and explains that the Ilonen shadow paging algorithm is similar to Kent. Ilonen, p. 1. Therefore, it would have been obvious to combine the teachings of Kent and Ilonen, thereby anticipating or rendering obvious the claims of the '211 patent.

3. U.S. Pat. No. 5,379,391 to Belsan et al. ("Belsan").

Belsan '391 teaches a method of maintaining a file system stored in a cache memory and on a disk storage system. *See* Fig.1. Belsan teaches at least a mapping table and a copy table. The mapping table (root inode) is maintained on disk and contains pointers to data blocks stored on disk. U.S. Pat. 5, 124,987, incorporated by reference, teaches that the mapping table is stored (backed up) on disk. '987 Pat. Col. 17:30-18:10. Pointers are disclosed in the Belsan specification at Col. 8:64-9:3, Figs. 2 and 3. One of ordinary skill in the art understands that the disclosed mapping table points to blocks that store a consistent state. Belsan further teaches that the mapping table is uploaded and modified in cache memory during write transactions, constituting an incore root inode. Contents of the on-disk mapping table are loaded into a cache hash table. *See e.g.* Col. 13:54-59. The disclosed file system uses copy-on-write, so that a block to be written is first copied into cache memory and then is re-written to a new location on disk. Col. 12:22-38; 13:42-14:6. As one of ordinary skill would readily appreciate, during a sequence of write operations, the cached mapping table and associated copy table point to some data buffers in cache (buffers that are currently being written) as well as some modified blocks The cached mapping table also points to unmodified blocks on disk, which are also pointed to by the un-updated mapping table stored on-disk. Therefore, Belsan anticipates or renders obvious at least claims 1, 9, and 17.

4. U.S. Pat. No. 5,218,695 to Noveck et al. ("Noveck").

Noveck discloses a file system storing data in memory and on disk. In particular, Noveck discloses an on-disk Unix inode, with direct and indirect blocks. Col. 3:29-35, 3:60-63. Noveck further teaches an in-core inode, which is an image of the on-disk inode, maintained in cache memory. Col. 6:25-31. As data in the file system is modified, the in-core structure differs from the on-disk inode and data. Col. 7: 14-24. The in-core inode points to blocks and buffers that have been modified since the on-disk inode was written. Therefore, Noveck anticipates or renders obvious at least claims 1, 9, and 17.

5. Gray et al., *The Recovery manager of the System R Database Manager*, ACM, 1981 (Gray).

Gray teaches a database system using shadowing (copy-on-write) to manage files. *See e.g.* Gray, Sec. 2.1, Fig. 7. A copy of a page table (equivalent to an inode) and data pages (equivalent to data blocks) as well as a directory root (equivalent to root inode) is cached in memory and resident on disk. Gray, Sec. 2.1, Sec. 2.7, Fig. 10. Gray teaches that the database takes periodic checkpoints, representing consistent system states by flushing the cache to disk and then writing to disk an updated directory root. Gray, Sec. 2.7, Fig. 10. For at least these reasons, Gray anticipates or renders obvious at least claims 1-3, 9-11, and 17-19. Furthermore, Gray teaches that the "directory root is duplexed on disk to tolerate failures while writing the directory root." Gray, Sec. 2.7. Therefore, Gray anticipates or renders obvious claims 4, 12, and 20.

II. STATEMENT POINTING OUT SUBSTANTIAL NEW QUESTION OF PATENTABILITY

Since claims 1-24 of the '211 Patent are not patentable over the prior art references cited above for the reasons set forth above, a substantial new question of patentability is raised for each claim. Further, these prior art references cited above are material to the subject matter of the '211 Patent. In particular, these prior art references provide teachings not provided during the prosecution of the '211 Patent. Therefore, a substantial new question of patentability has been raised, and reexamination is respectfully requested.

CONCLUSION

Based on the above remarks, it is respectfully submitted that a substantial new question of patentability has been raised with respect to Claims 1-24 of the '211 Patent. Therefore, reexamination of Claims 1-24 is respectfully requested.

Any fee due for this reexamination may be charged to Deposit Account No. 07-1896.

Respectfully submitted,

DLA PIPER US LLP

Date: December 14, 2007

By: Ronald K. Yin
Ronald L. Yin
Reg. No. 27,607

Attorneys for Applicant(s)

Ronald L. Yin
DLA Piper US LLP
2000 University Avenue
East Palo Alto, CA 94303-2248
650-833-2437 (Direct)
650-833-2000 (Main)
650-833-2001 (Facsimile)
ronald.yin@dlapiper.com

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 2

Complete if Known	
Patent Number	6,882,211
Issue Date	May 10, 2005
First Named Inventor	Hitz et al.
Art Unit	N/A
Examiner Name	N/A
Attorney Docket Number	347155-29

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No [1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US- 5,379,391	01-03-1995	Belsan et al.	
		US- 5,218,695	06-08-1993	Noveck	
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] "Number[4] "Kind Code[5] *(if known)*	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional) [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04 [3]Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4]For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST. 16 if possible [6]Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449B/PTO	Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Patent Number	6,892,211
	Issue Date	May 10, 2005
	First Named Inventor	Hitz et al.
	Art Unit	N/A
(Use as many sheets as necessary)	Examiner Name	N/A
Sheet 2 of 2	Attorney Docket Number	347155-29

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Quinlan, *A Cached WORM File System*, Software – Practice And Experience, Vol. 21(12), 1289-1299, December 1991 ("Quinlan").	
		Popek, Walker, *The LOCUS Distributed System Architecture*, MIT Press, Cambridge, Mass., 1985 ("Popek").	
		Ylonen, *Concurrent Shadow Paging: A new Direction for Database Research*, Helsinki University of Technology, TKO-B86, 1992 ("Ylonen").	
		Margo Ilene Seltzer , *File System Performance and Transaction Support*, Doctoral Dissertation, UC Berkeley, 1992 ("Seltzer").	
		Schilling, *Design and implementation of a fast file system for Unix with special consideration of technical parameters of optical storage media and multimedia applications*, Thesis submitted to Technical University of Berlin on 5/23/1991, translated from German. ("Schilling"). All pages cited are to the English translation.	
		Leffler, McKusick, et. al., *4.3BSD Unix Operating System*, Addison-Wesley Publishing Co., 1990 ("Leffler").	
		Bach, *The Design of the Unix Operating System*, Prentice Hall, 1990 ("Bach").	
		Rosenblum, Ousterhout, *The LFS Storage Manager*, Computer Science Division, Electrical Engineering and Computer Sciences, University of California, Summer '90 USENIX Technical Conference, Anaheim, California, June 1990. ("Rosenblum").	
		Rosenblum, Ousterhout, *The Design and Implementation of a Log-Structured File System*, Proceedings of the 13th ACM Symposium on OS Principles, 1991 ("Rosenblum and Ousterhoot").	
		Kent, *Performance and Implementation Issues in Database Crash Recovery*, Ph.D. Dissertation, Princeton University, 1985 ("Kent").	
		Gray et al., *The Recovery Manager of the System R Database Manager*, ACM, 1981 ("Gray").	

Examiner Signature			